2 Types of Grammars

Type 0: no restrictions

Type 1 (context sensitive): only two types of rules allowed

- $lAr \to lwr$ where
  - $l, r \in V^*$ [so they can be anything, including empty]
  - $A \in N$ [so A is a single non-terminal]
  - $w \in V^+$ [That is, $w \neq \lambda$.]
  - shorthand: `[left][nonterminal][right]` $\to$ `[left][non-empty][right]`
- $S \to \lambda$
  - shorthand: `[start]` $\to$ `[empty string]`
  - **If this rule is used, the the start symbol S may not appear on the right side of any rule.**

Type 2 (context free): only one type of rule allowed

- $A \to w$ where
  - $A \in N$ [so A is a single nonterminal symbol]
  - $w \in V^*$ [so no restriction here]
- shorthand: `[nonterminal]` $\to$ `[any string]`

Type 3 (regular): Three types of rules allowed

- `[nonterminal]` $\to$ `[terminal] [non terminal]` [Example: $A \to bC$]
- `[nonterminal]` $\to$ `[terminal]` [Example: $A \to b$]
- `[start]` $\to$ `[empty string]` [Example: $S \to \lambda$]

A regular/context free/context sensitive language is a language that can be generated by a regular/context free/context sensitive grammar.

1. For Grammars 1 - 4 from the previous section, determine whether it is regular, context free, or context sensitive. (Some grammars will be more than one. All grammars are type 0.)
2. Consider **Grammar 8** (G_8): alphabet: $\{A, B, C, x, y, z\}$, terminals: $\{a, b, c\}$, start symbol: A, production rules:
   - $A \to \lambda$
   - $A \to xBC$
   - $BC \to xyC$
   - $B \to x$
   - $B \to AC$
   - $xCx \to zyz$
   - $C \to Ax$
   - $C \to yB$
   - $C \to aBc$
   - $C \to z$

   a. Why isn't this a type 1 grammar?
   b. Which rules can you eliminate to make this a type 1 grammar? (Eliminate as few as possible.)
   c. Which rules can you eliminate to make this a type 2 grammar? (Eliminate as few as possible.)
   d. Which rules can you eliminate to make this a type 3 grammar? (Eliminate as few as possible.)
3. True or false: Every grammar of type n is a grammar of type $n - 1$.
4. True or false: Every language of type n is a language of type $n - 1$.

2.1 Backus-Naur Form (BNF) for Context Free Grammars

Shorthand notation for type 2 grammars.

- nonterminals denoted with $\langle\rangle$.
- $\rightarrow$ written as `::=`
- All rules with same nonterminal on left written together with the list of possible right sides separated by `|`.

Example: If we have production rules $A \rightarrow Aa$, $A \rightarrow a$, and $A \rightarrow AB$, we will write

```
<A> ::= <A>a | a | <A><B>
```

BNF for ALGOL 60 identifier

```
<identifier> ::= <letter> | <identifier><letter> | <identifier><digit>
<letter> ::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

BNF for signed integer

```
<signed integer> ::= <sign><integer>
<sign> ::= + | -
<integer> ::= <digit> | <digit><integer>
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

BNF specification for Java

You can find BNF for various programming languages online. For example: https://users-cs.au.dk/amoeller/RegAut/JavaBNF.html has a BNF specification for Java.

Exercises

5. Why does BNF notation only work for context free grammars?
6. For each context free grammar we have seen, give its BNF representation.